// C++ code to reverse a

// stack using recursion

#include<bits/stdc++.h>

using namespace std;

// using std::stack for

// stack implementation

stack<char> st;

// intializing a string to store

// result of reversed stack

string ns;

// Below is a recursive function

// that inserts an element

// at the bottom of a stack.

char insert\_at\_bottom(char x)

{

    if(st.size() == 0)

    st.push(x);

    else

    {

        // All items are held in Function Call

        // Stack until we reach end of the stack

        // When the stack becomes empty, the

        // st.size() becomes 0, the above if

        // part is executed and the item is

        // inserted at the bottom

        char a = st.top();

        st.pop();

        insert\_at\_bottom(x);

        // push allthe items held in

        // Function Call Stack

        // once the item is inserted

        // at the bottom

        st.push(a);

    }

}

// Below is the function that

// reverses the given stack using

// insert\_at\_bottom()

char reverse()

{

    if(st.size()>0)

    {

        // Hold all items in Function

        // Call Stack until we

        // reach end of the stack

        char x = st.top();

        st.pop();

        reverse();

        // Insert all the items held

        // in Function Call Stack

        // one by one from the bottom

        // to top. Every item is

        // inserted at the bottom

        insert\_at\_bottom(x);

    }

}

// Driver Code

int main()

{

    // push elements into

    // the stack

    st.push('1');

    st.push('2');

    st.push('3');

    st.push('4');

    cout<<"Original Stack"<<endl;

    // print the elements

    // of original stack

    cout<<"1"<<" "<<"2"<<" "

        <<"3"<<" "<<"4"

        <<endl;

    // function to reverse

    // the stack

    reverse();

    cout<<"Reversed Stack"

        <<endl;

    // storing values of reversed

    // stack into a string for display

    while(!st.empty())

    {

        char p=st.top();

        st.pop();

        ns+=p;

    }

    //display of reversed stack

    cout<<ns[3]<<" "<<ns[2]<<" "

        <<ns[1]<<" "<<ns[0]<<endl;

    return 0;

}